VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozpoznání dopravního značení z obrazů

Traffic Signs Recognition from Images

2019 Petr Bednář

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Bednář**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Rozpoznání dopravního značení z obrazů
Traffic Signs Recognition from Images

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je detekce a rozpoznání dopravního značení v obrazech. Využijte dostupných frameworků pro strojové učení k rozpoznání dopravního značení např. z fotografií podobných Google StreetView.

Ve své práci proveďte:
1. Nastudujte možnosti detekce a klasifikace objektů s využitím strojového učení.
2. Vybranou metodu detekce využijte pro natrénování příslušného detektoru a klasifikátoru.
3. Vaši implementaci řádně otestujte na vhodném datasetu.
4. Zhodnoťte dosažené výsledky v závěru práce.

Seznam doporučené odborné literatury:

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg: SSD: Single Shot MultiBox Detector, https://arxiv.org/abs/1512.02325, 2015

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29.dubna 2019

P. Bezlnor

.....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 29.dubna 2019 ..... P. Bednář .....

Rád bych na tomto místě poděkoval Ing. Janu Gaurovi, Ph.D. za odborné vedení a rady při zpracování této práce. Rovněž bych chtěl poděkovat rodině za podporu při studiu a také přítelkyni za její trpělivost se mnou při zpracování této práce.

Abstrakt

Cílem této práce bylo nastudovat možnosti detekce a klasifikace objektů s využitím strojového učení a následně implementovat detektor pro dopravního značení v obrazech. Na základě zvážení výběru frameworku, byla implementace napsána ve frameworku Tensorflow, s využitím Object detection API. Následně byl vytvořen soubor dat pro trénování a testování neuronové sítě. Nakonec byly porovnány a vyhodnoceny výsledky rozdílných modelů pro objektové rozpoznávání.

Klíčová slova: Strojové učení, Tensorflow, Object detection API, Python, počítačové vidění, Faster R-CNN, SSD, dopravní značky, model

Abstract

The aim of this thesis was to study the possibilities of detection and classification of objects using machine learning and implementation of detector for traffic signs in images. Based of framework selection, the implementation was written in Tensorflow framework, using Object detection API. Then a dataset for training and testing of the neural network was created. Finally, the results of different models for object recognition were compared and evaluated.

Key Words: Machine learning, Tensorflow, Object detection API, Python, computer vision, Faster R-CNN, SSD, traffic signs, model

Obsah

Seznam použitých zkratek a symbolů

CNN	–	Convolutional Neural Network
R-CNN	–	Region-based Convolutional Neural Networks
R-FCN	–	Region-based Fully Convolutional Networks
YOLO	–	You Only Look Once
SSD	–	Single Shot Detector
IoU	–	Intersection over Union
mAP	–	Mean Average Precision
RoI	–	Region of Interest
COCO	–	Common Objects in Context
RPN	–	Regional Proposal Network
FPS	–	Frames Per Second
SVM	–	Support Vector Machines
SPP	–	Spatail Pyramid Pooling
VOC	–	Visual Object Classes
API	–	Application Programming Interface
MLP	–	Multilayer Perceptron
BP	–	Back Propagation
HSI	–	Hue, Saturation, Intensity

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V současné době jsou hluboké neuronové sítě v klasifikaci obrazu lepší než lidé, což ukazuje, jak hodně je tato technika mocná, ale musíme vzít v ohled, že my lidé děláme mnohem víc věcí než pozorování netřídíme jen věci do určitých kategorii. To nám však nebrání tomu, aby nám tato technika usnadnila život.

Rozpoznání dopravních značek je velmi zajímavou oblastí v počítačovém vidění. Důvodem je možnost využití těchto systému v automobilovém průmyslu, například pro informování nebo kontrolu řidiče o rychlostním limitu nebo jen zobrazovat řidiči užitečné informace o podmínkách na silnici.

V této práci bude popsán stručný úvod do problematiky strojového učení a typy algoritmu pro detekci objektů, také se podíváme na již existující řešení pro detekci dopravních značek. Kromě toho budou popsány použité knihovny a nástroje, které byli využity. Rovněž zde bude ukázka toho, jak si připravit data pro trénování a testování modelu a jak probíhá vytvoření vlastního modelu. Závěrem bude porovnání výsledků, různých typů detekčních sítí.

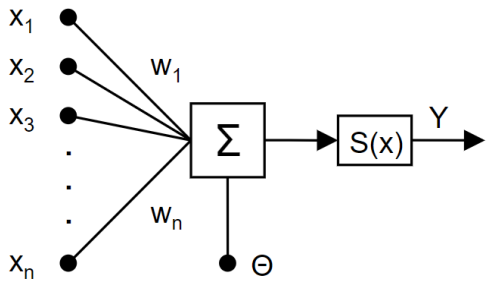
2 Strojové učení

Strojové učení je věda, která se zabývá programováním počítačů tak, aby byly schopné se učit z dat. Strojové učení můžeme použít téměř ve všech odvětvích života a práce, ale některé jsou více známé jako je počítačová vidění, rozpoznávání řeči, překlad cizích jazyků a zdravotnictví. Jako příklad strojového učení může být spam filtr pro emaily, který se může naučit rozpoznávat spam emaily, poté co antispam programu předložíme příklady spam emailů a běžných emailů (no spam). Příkladům s jejichž pomocí se program naučí rozpoznávat spam emaily se nazývají trénovací data. Každý jednotlivý příklad nazýváme trénovací instance nebo vzorek [1].

Strojové učení se hodí pro problémy, které vyžadují velké množství ručního ladění nebo dlouhé seznamy pravidel. Jeden algoritmus strojového učení může často velmi zjednodušit kód a zároveň dosáhnout lepších výsledků. Strojové učení je vhodné například i pro komplexní problémy, kde je velmi obtížné používat běžné programovací techniky. Rovněž může být použito v prostředí, které není stabilní a systém se může přizpůsobovat novým datům [1].

2.1 Umělá neuronová síť

Umělé neuronové sítě patří do oboru umělé inteligence, které se inspirují strukturou lidské nervové soustavy. Základním prvkem přirozené i umělé neuronové sítě je neuron neboli perceptron. Neurony jsou navzájem propojeny a předávají si signály. Platí přitom, že každý neuron může mít více vstupů, ale jen jeden výstup (tento výstup ale může být poslán i více než jednomu dalšímu neuronu). Vstupem neuronu může být buď výstup z jiného neuronu nebo informace z vnějšku. Každý spoj je ohodnocen vahou, která modifikuje intenzitu procházejícího signálu [2].



Obrázek 1: Model umělého neuronu

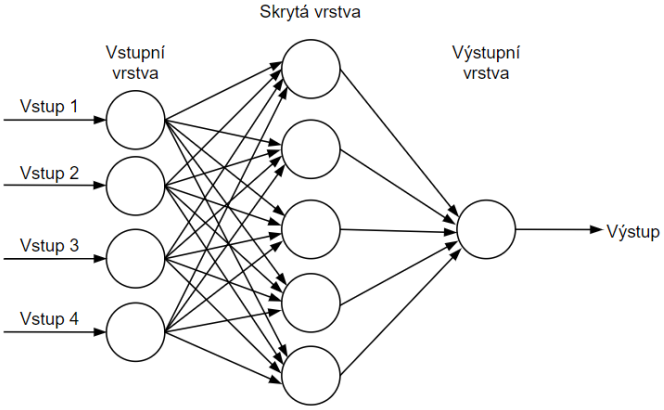
Výstup neuronu tohoto typu lze matematicky vyjádřit:

$$Y = S\left(\sum_{i=1}^{N}(w_i x_i) + \Theta\right) \tag{1}$$

kde Y je výstup neuronu, x_i jsou vstupní neurony, w_i jsou váhy spojené s předchozím neuronem, Θ je citlivost neuronu, $S(x)$ je přenosová funkce neuronu.

2.2 Dopředná neuronová síť

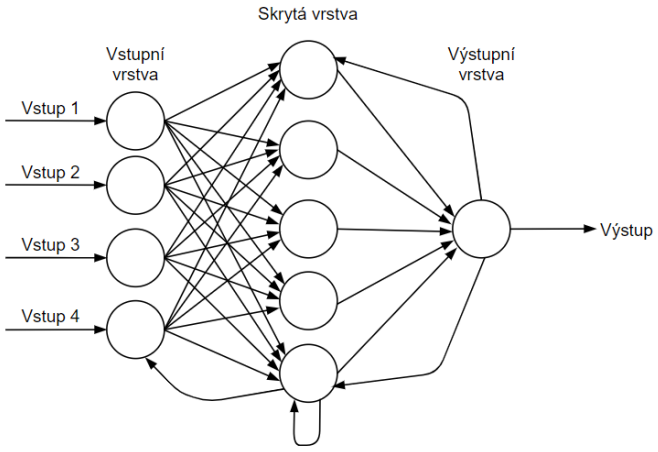
Mezi základní typ neuronové sítě, patří dopředná neuronová síť, která se používá pro analýzu a generování obrazu. Dopředná je proto, že se šíří od vstupu jednosměrně směrem k výstupu sítě. Síť má vstupní a výstupní vrstvu a většinou implementujeme pouze s jednou, eventuálně s dvěma skrytými vrstvami.



Obrázek 2: Dopředná neuronová síť

2.3 Rekurentní neuronová síť

Rekurentní neuronová síť je dalším typem neuronové sítě. Signál v neuronové síti neproudí jen jedním směrem, ale může se vracet nebo točit ve smyčce. Takové sítě mají jednu zásadní vlastnost a tou je možnost uchování informace. Jinými slovy, rekurentní sítě mají paměť [3].



Obrázek 3: Rekurentní neuronová síť

2.4 Konvoluční neuronová síť

Konvoluční neuronová síť je algoritmus hlubokého učení, který může přijímat vstupní obraz, přiřazovat důležitost (váhy a biases) různým aspektům/objektům v obraze a být schopen je odlišit od jiných. Předběžné zpracování požadované v ConvNet je mnohem nižší ve srovnání s jinými klasifikačními algoritmy. Zatímco v primitivních metodách jsou filtry konstruovány ručně, s dostatečným tréninkem, ConvNets mají schopnost naučit se tyto filtry/charakteristiky sami.

Architektura ConvNet je analogická architektura struktury neuronů v lidském mozku a byla inspirována organizací Visual Cortex. Jednotlivé neurony reagují na podněty pouze v omezené oblasti vizuálního pole známého jako recepční pole. Sbírka takových polí se překrývá, aby pokryla celou vizuální oblast [4].

2.5 Strategie učení neuronových sítí

Používáme dvě hlavní strategie učení neuronových sítí, a to učení s učitelem a učení bez učitele. V této práci se budu zabývat pouze učením s učitelem.

2.5.1 Učení s učitelem

V učení s učitelem jsou poskytovány jak vstupy, tak i výstupy. Trénování probíhá iterativně, kdy algoritmus jednotlivé prvky trénovací množiny předkládá postupně neuronu, zjišťuje jeho odezvu na předložený vstup a na základě odchylky jeho výstupu od výstupu požadovaného provádí korekci vah neuronu. Interval, ve kterém dojde k předložení všech vzorů trénovací množiny alespoň jednou, nazýváme epochou učení. K naučení sítě může být dle komplexnosti problému zapotřebí desítky až tisíce epoch [2].

2.5.2 Učení bez učitele

Pro učení bez učitele je typické, že adaptační algoritmus nemá k dispozici žádné kritérium správnosti hledané transformace vstupních dat. Pracuje na principu shlukování, kdy ve vstupním prostoru dat hledá „sobě podobné“ elementy. Na základě předkládaných vzorků vstupních dat provádí jejich třídění do skupin, bez možnosti posouzení správnosti zatřídění. Počet hledaných skupin může být předem dán [2].

3 Algoritmy pro detekci objektů

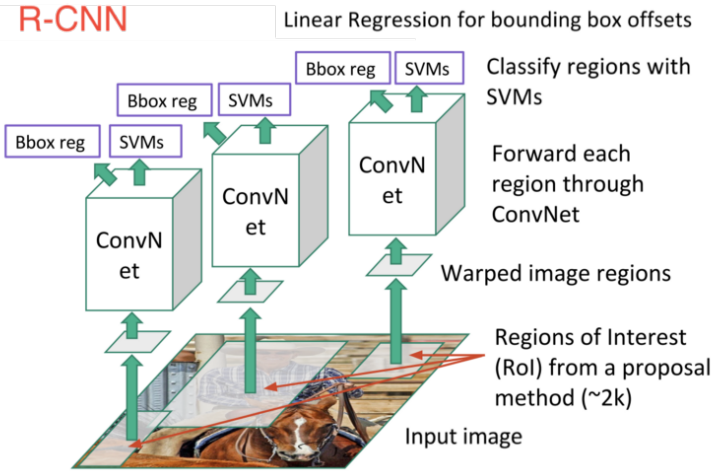
V této kapitole budou popsány aktuálně velmi populární typy detekčních algoritmu. Ve všech případech se jedná o neuronové sítě. U jednotlivých algoritmů se pozastavíme a přiblížíme si jejich funkčnost a klíčové vlastnosti, také vezmeme v potaz jejich přesnost a rychlost detekce. Tohle je velmi důležitá část, díky které můžeme následně vybrat vhodného detekčního algoritmu pro rozpoznávání dopravního značení.

3.1 Region-based Convolutional Neural Network (R-CNN)

R-CNN je jedním z moderních přístupů detekce objektu založená na CNN, který je schopen lokalizovat a detekovat objekty v obrazech. Výstupem je sada ohraničujících boxů, které odpovídají každému z detekovaných objektů, stejně jako výstup třídy pro každý detekovaný objekt. Doposud byly zveřejněný 4 verze R-CNN architektur: R-CNN, Fast-RCNN, Faster-RCNN a R-FCN, kde každá nová verze přináší určitá zlepšení rychlosti a přesnosti sítě oproti předešlé verzi, až na Faster-RCNN a R-FCN, které jsou rovnocenné v rámci úspěšnosti. Cílem bylo dosáhnout detekce objektů v reálném čase, nicméně ani jedna z těchto architektur nedokáže detekovat objekty v reálném čase.

3.1.1 R-CNN

R-CNN [11] lze shrnout do 3 kroků. Začne procházet obrázek pro možné kandidáty pomocí algoritmu selektivního vyhledávání, kde vygeneruje okolo 2000 potencionálních oblastí. Následně spustí CNN pro každou navrhovanou oblast. Každá navrhovaná oblast je transformována, aby měla pevnou velikost, jak vyžaduje CNN. Poté vezme každý výstup z CNN a vloží jej do SVM, aby byla oblast klasifikována, a do lineárního regresoru pro zlepšení původního ohraničujícího boxů, pokud takový objekt existuje.



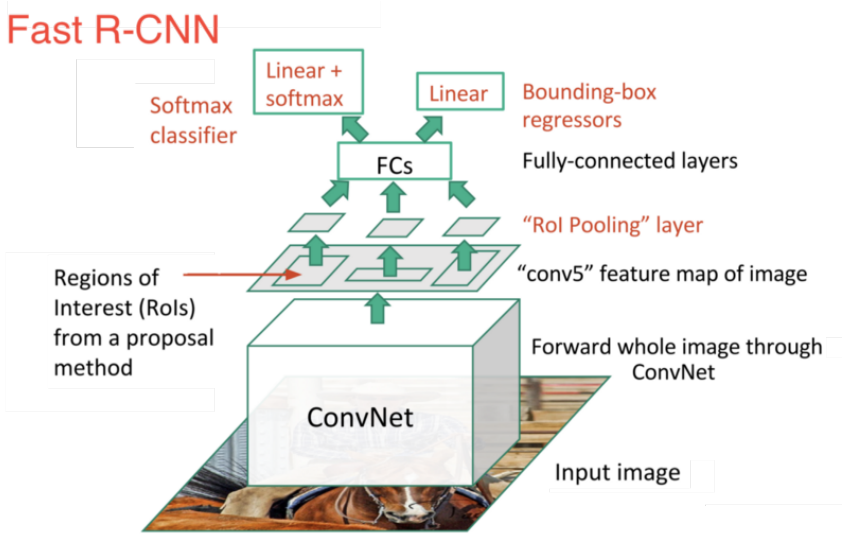
Obrázek 4: Model R-CNN [5]

R-CNN má velkou nevýhodu, a tou je rychlost, kde detekce na obrázku 600 × 1000 pixelů trvá 50 sekund, a nejen rychlost detekce, ale i rychlost trénování je velmi nákladná na čas. Další nevýhodou je použití SVM, které nijak neovlivní váhy konvoluční neuronové sítě. V neposlední řadě SVM a regresní trénování ohraničujících boxu, kde jejich funkce jsou extrahovány z každého návrhu objektu z každého obrázku a poté zapsány na disk, což vyžaduje stovky gigabytů na disku. Nicméně úspěšnost detektoru dosáhla 53,7 mAP na datové sadě Pascal VOC 2010, to bylo o více než 25% lepší průměrná přesnost než ostatní metody.

3.1.2 Fast R-CNN

Verze Fast R-CNN [11] se podobá v mnoha ohledech R-CNN, ale zlepšila se jeho rychlost detekce, to díky dvou hlavním rozšířením. Nahrazením SVM vrstvou softmax, čímž se rozšíří neuronová sít o předpovídání namísto vytváření nového modelu. Druhým rozšířením je spuštěním jedné CNN přes celý obraz, namísto spouštění 2000 CNN přes 2000 překrývajících se oblastí.

RoI pooling layer je speciálním případem vrstvy prostorové pyramidy, která je zavedena ve Spatial Pyramid Pooling (SPP), pro vizuální rozpoznávání. Hlavní funkcí vrstvy RoI je přetvořit vstupy s libovolnou velikostí na výstup s pevnou délkou z důvodu omezení velikosti v plně propojených vrstvách. Hlavním účelem takového RoI pooling layer je urychlení tréninkového a zkušebního času a také trénování celého systému end-to-end.

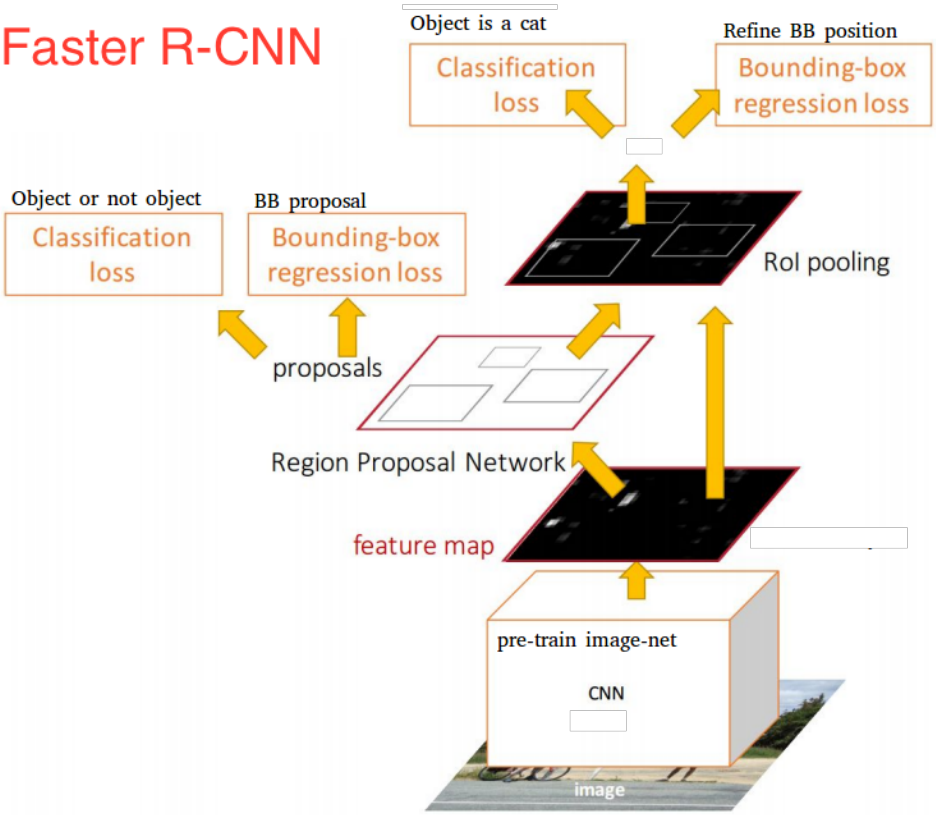


Obrázek 5: Model Fast R-CNN [5]

Vylepšení u Fast R-CNN zrychlila 25násobně detekci oproti R-CNN, což u obrázku 600×1000 pixelů odpovídá 2 sekundám a také došlo k drobnému zvýšení úspěšnosti na 65.7 mAP u datové sady Pascal VOC 2012.

3.1.3 Faster R-CNN

Verze Faster R-CNN [6] se skládá ze dvou sítí. Jedna síť je určená pro návrh oblastí, jde o novou komponentu, kterou nazýváme RPN. Druhá síť je určena k detekci objektů z daných návrhů. RPN nahrazuje algoritmus selektivního vyhledávání a je mnohem rychlejší a přesnější. Zlepšení je zapříčiněno díky schopnosti, učení klasifikace objektů pozadí a popředí. RPN lze popsat jako plně konvoluční neuronovou síť. Vyhledávání vhodných oblastí je velkým problémem předchozích verzi, zatím co u Faster R-CNN díky RPN byla výrazně zvýšena efektivita vyhledávání objektů, kde namísto 2000 oblastí u R-CNN je jich u Faster R-CNN nutné prohledat jen 300 a tím zároveň zrychlit celý proces detekce.

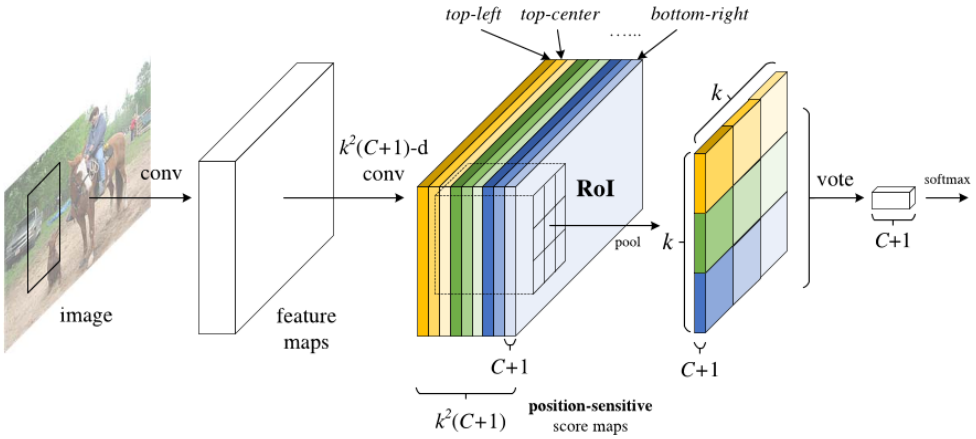


Obrázek 6: Model Faster R-CNN [5]

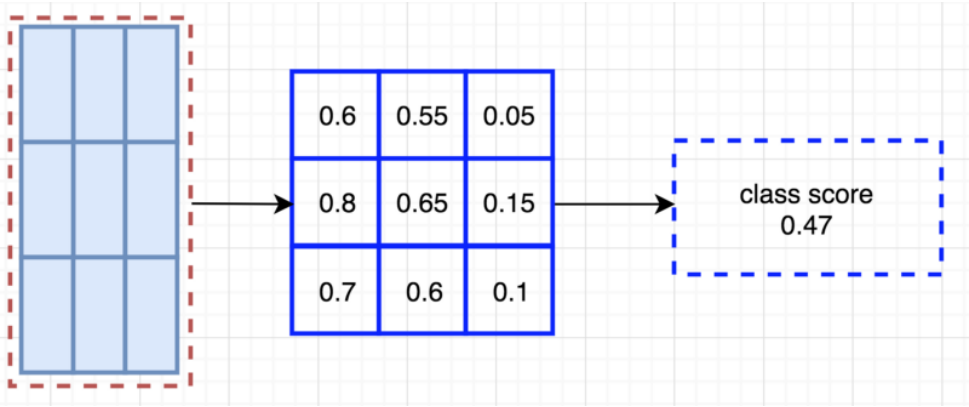
Faster R-CNN díky komponentě RPN je 10násobně rychlejší než Fast R-CNN, což u obrázku 600×1000 pixelů odpovídá 0,2 sekundám a rovněž došlo ke zvýšení úspěšnosti na 67.0 mAP u datové sady Pascal VOC 2012.

3.1.4 R-FCN

Verze Region-based Fully Convolutional Network (R-FCN) [5] je plně konvoluční síť, sdílí 100% výpočtů přes každý jednotlivý výstup, což znamená, že celý výpočet je sdílen v celé síti. Nejprve spustí CNN, přes vstupní obraz. Přijde přidání plně konvoluční vrstvy pro vytvoření score bank. Mělo by existovat $k^2(C+1)$ score bank, kde k^2 představuje počet relativních pozic pro rozdělení objektů. Spustí se plně konvoluční síť pro návrh regionů (RPN) pro generování oblastí zájmů. Každé RoI je rozděleno na stejných k^2 subregionu jako score maps. V každém subregionu zkontroluje score bank, aby zjistil zda subregion odpovídá pozici určitého objektu. Jakmile je každému subregionu přiřazena hodnota pro každou třídu, zprůměruje subregiony k získání výsledného skóre pro třídu. Klasifikuje RoI softmax nad zbývajícím $C+1$ rozměrným vektorem.



Obrázek 7: Model R-FCN [5]



Obrázek 8: Průměr subregionů pro získání skóre třídy [7]

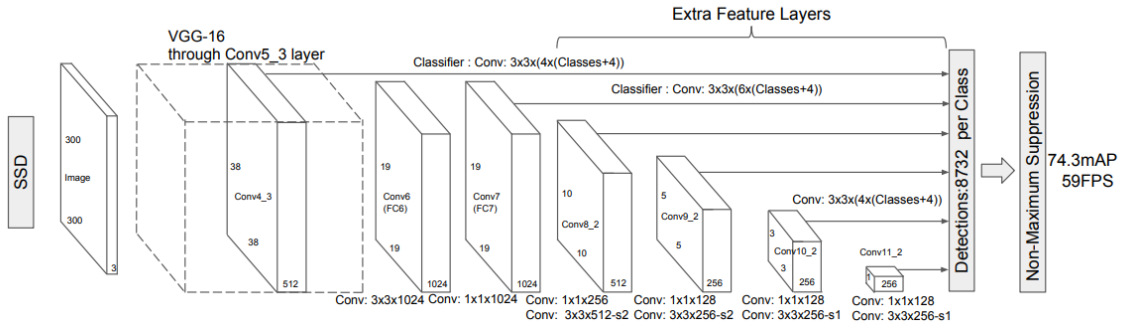
R-FCN je 8násobně rychlejší než verze Faster R-CNN a také přináší zvýšení úspěšnosti detekce na datové sadě PASCAL VOC na 79 mAP.

3.2 Single Shot MultiBox Detector (SSD)

Single Shot MultiBox Detector (SSD) [12] je navržen pro detekci objektů v reálném čase. Přístup SSD je založen na dopředné konvoluční síti, která vytváří kolekci ohraničení boxů s danými rozměry a jejich ohodnocením pravděpodobnosti pro výskyt objektů v těchto boxech. Následně dojde k potlačení nemaxim (non-maximum suppression) pro získání výsledné detekce.

První vrstva je založená na standardní architektuře používanou pro vysoce kvalitní klasifikaci obrazu, kterou je VGG16. Za vrstvy základní sítě jsou přidány konvoluční funkční vrstvy. Tyto vrstvy se postupně zmenšují a umožňují predikovat detekci ve více měřítcích. Konvoluční model pro predikci detekce se liší pro každou funkční vrstvu.

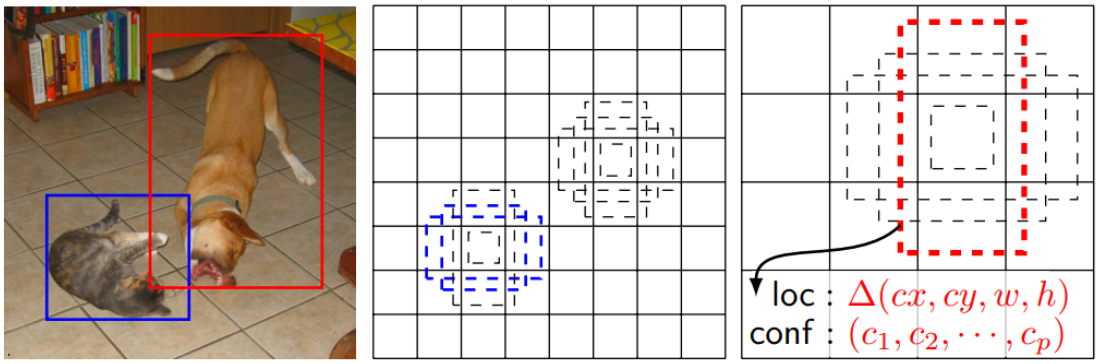
SSD nepoužívá RPN, místo toho to řeší velmi jednoduchou metodou. Vypočítá jak lokaci, tak klasifikaci použitím malých konvolučních filtrů. Modul starající se o detekci pracuje s odlišnými vrstvami příznaků, což zajišťuje vysoký počet predikcí. Vrstvy příznaků se rozdělí do mřížky. Na každé buňce v této mřížce je predikován posun vzhledem k výchozímu ohraničení, stejně jako skóre pro jednotlivé třídy, které indikují přítomnost instance třídy v každé z těchto buňkách.



Obrázek 9: Schéma modelu SSD300. Do sítě vstupuje obraz o rozměrech 300×300 pixelů. Ten je předán VGG16 architektura. Na konci modelu jsou přidány různé vrstvy příznaků, které umožňují predikce u různých měřítek.[12]

SSD má dva typy modelů, pracující s různými velikostmi vstupních obrazu. SSD300 pracující se vstupním obrazem 300×300 pixelů a SSD512 pracující se vstupním obrazem 512×512 pixelů.

SSD je ve srovnání s Faster R-CNN mnohem rychlejší. Na datové sady Pascal VOC 2007, SSD300 dosáhlo úspěšnosti 74,3 mAP s rychlosti detekce 46 FPS a SSD512 dosáhlo 76,8 mAP s rychlosti detekce 19 FPS.



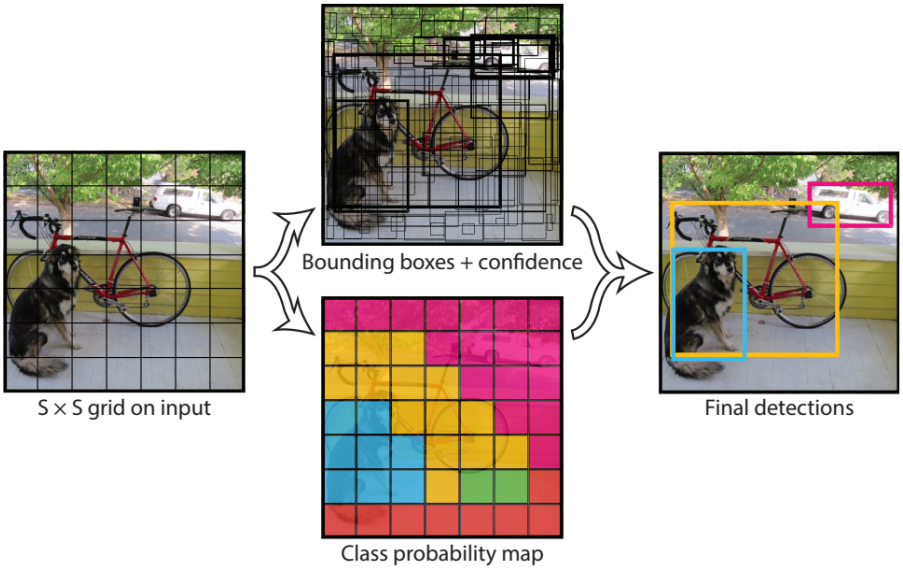
Obrázek 10: Princip SSD modelu. [12]

3.3 You Only Look Once (YOLO)

You Only Look Once (YOLO) [13] na rozdíl od ostatních klasifikačních řešení, které přistupují ve dvou fázích a to vyhledávání zájmových oblastí a následné rozpoznání těchto oblastí, přistupuje jen pomocí jedné neuronové sítě, která vyhledává zájmové oblasti a zároveň klasifikuje objekty. Díky tomu, že je celý proces proveden nad jednou neuronovou sítí, je mnohem méně složitý na výpočet. Při detekci algoritmus rozdělí vstupní obraz na mřížku o velikosti $S \times S$. Jestliže je objekt ve středu dané buňky, je pak tato buňka zodpovědná za jeho detekci. Každá z buněk předpovídá B ohraničujících boxů a pro každý ohraničující box jeho skóre, které určuje, jestli ohraničující box obsahuje daný objekt. Současně definuje jak moc přesný je ohraničující box kolem objektu. Bude-li skóre nulové, v dané oblasti se nenachází žádný objekt. Pokud se v oblasti objekt nachází, bude skóre rovno poměru průniku a sjednocení plochy ohraničujícího boxů buňky, neboli Intersection over Union (IoU).

$$IoU = \frac{\textit{společná oblast}}{\textit{veškerá oblast}}$$

kde *společná oblast* odpovídá překrytí predikovaného ohraničujícího boxu a anotovaného ohraničujícího boxu, *veškerá oblast* odpovídá oblasti, která zahrnuje jak predikovaný ohraničující box, tak anotovaný ohraničující box. [14]



Obrázek 11: Model detektoru YOLO rozdělí vstupní obraz na mřížku s $S \times S$ buňkami a pro každou buňku vytvoří B hraničních oblastí, jistotu správnosti ohraničení a vektor predikcí tříd C. Výsledný tenzor je pak vyjádřen vztahem: $S \times S \times (B * 5 + C)$[13]

Každý ohraničující box se skládá z 5 předpovědí: x, y, w, h a skórem jistoty detekce. Kde souřadnice x, y vyznačují relativní střed oblasti vzhledem k hranicím buňky. Šířka a výška jsou předpovídány vzhledem k celému obrazu. Skóre jistoty detekce je vyjádřeno díky IoU. Každá buňka pravidelné mřížky následně definuje pravděpodobnost třídy objektů dosažené v buňce. Pro každou buňku je definován pouze jeden vektor predikce a to nezávisle na počtu ohraničených boxů. YOLO dokáže pracovat rychle a to 45 FPS s přesností 64 mAP. Avšak YOLO pracuje dobře, především při detekci velkých objektu, zatímco má problém s detekci malých objektů, proto nebyl vybrán pro implementaci detektoru pro rozpoznávání dopravních značek v obrazech.

4 Existující řešení detekce značek

Pro řešení problematiky detekce a klasifikace dopravního značení, existují již desítky řešení, které na tuto problematiku využívají nejrůznější algoritmy. Většina těchto prací si klade dost podobné cíle, a tím je detektor a klasifikátor dopravního značení s velmi vysokou přesností a možností zpracování snímků v reálném čase.

4.1 Real-time Detection and Recognition of Traffic Signs

Jde o konferenční příspěvek publikovaný v červnu 2010, na kterém se podílel A. Martinović, G. Glavaš, M. Juribašić, D. Sutić a Z. Kalafatić z Faculty of Electrical Engineering and Computing v Zagrebu [15]. Cílem práce bylo vytvořit detektor a rozpoznávač dopravního značení v reálném čase, avšak pouze se zaměřením na 5 nejběžnějších značek, které byly podobného vzhledu. Pro detekci objektů v obraze se rozhodli využívat Viola-Jones algoritmus a to z důvodu fungování v reálném čase. Detektor Viola-Jones pracuje principem posunutím detekčního okna přes obraz. Zlepšení klasifikátoru bylo provedeno AdaBoost. Na trénování klasifikátoru bylo použito 757 pozitivních obrázků, které byly oříznuty a normalizovány na velikost 24×24 pixelů a 3000 obrázků bylo použito jako negativ. Pro trénování kaskád bylo použito OpenCV a trénování trvalo přibližně 16 hodin na 4 CPU s povoleným OpenMP. Trénovaná kaskáda byla následně testována na testovací sadě o 246 obrazech, které byly vytvořený různými kamerami za různých světelných podmínek.

Tabulka 1: Výsledek experimentu pro Viola-Jones detektor [15]

Měřítko	Nalezeno	Nenalezeno	Falešně pozitivní
1.3	61.53%	38.46%	11.88%
1.2	67.13%	32.86%	18.88%
1.1	75.17%	24.83%	28.67%

Zmenšení velikosti způsobilo zvětšení šance, že značka bude detekována, ale také zvýšilo potřebný čas pro dokončení algoritmu. V práci bylo použito měřítko 1,1, protože poskytovalo nejlepší poměr mezi úspěšností a rychlosti snímku (nad 20 FPS). Po úspěšné detekci značky v obraze, začíná klasifikační proces. Klasifikátor očekává odpovídající vstupní vektor, musíme nejprve připravit pomocí předzpracování obrazu. Protože výsledná značka z detekční fáze může mít libovolnou velikost a aby bylo možné ji klasifikovat je potřeba velikost normalizovat. V práci byla použila standardní velikost značky 10×10 pixelů. Změna velikosti byla implementována pomocí bilineárního interpolačního algoritmu. Poté je značka převedena do stupně šedi a transformována do binárního obrazu z použití prahovacího algoritmů.

Pro klasifikaci byla použita neuronová síť. Byl zvolen typ vícevrstvý percepton (MLP) s metodou backpropagation (BP) pro trénování klasifikátoru. Pro projekt byla vytvořená vlastní softwarová knihovna pro MLP trénovaného podle BP algoritmu. Byly vytvořeny a vytrénovány dvě knihovny MLP s různými vícevrstvými perceptony. Účelem první sítě bylo klasifikovat a

zařadit dopravní značku do jedné z pěti kategorii. Úkolem druhé MLP bylo rozpoznat aktuální rychlostní limit, pokud první síť klasifikuje vstupní vektor do kategorie rychlostních limitů. Druhá síť byla vycvičena k rozpoznání číslic (0-9).

Tabulka 2: Počet neuronu pro jednotlivé vrstvy [15]

	Vstupní vrstva	Skrytá vrstva	Výstupní vrstva
Obecné značky MLP	100 (10×10 pixelů)	10	5
Rychlostní limity MLP	72 (6×12 pixelů)	10	10

Tabulka 3: Faktory učení korekční váhy, maximální počet epoch a průměrných chyb [15]

	Faktor učení (η)	Maximální počet epoch	Průměrná vyhovující chybovost epoch
Obecné značky MLP	ze začátku 0.1, 0.05 ,když chybovost klesle pod 0.1	10 000	0.01
Rychlostní limity MLP	ze začátku 0.01, 0.005 ,když chybovost klesle pod 0.1	50 000	0.001

Vstupní trénovací vzorky dopravních značek pro první síť, byly získány lokalizačním procesem na počátečních snímcích o velikosti 10×10 pixelů. Vstupní trénovací vzorky pro rychlostní limity byly o velikosti 6×12 pixelů, obraz čistého decimálního čísla a jejich kopie s náhodně přidaným šumem, překlopením 10% bitů v originálu binárního obrazu. Systém využívá auto-degrading princip posilování, pro sledování značek.

Výstupem práce je systém fungující na popsaných principech s dvěma front-end aplikacemi se stejným jádrem. Jeden pro detekci a rozpoznání dopravního značení v obrazech, druhý pro vstupní soubor typu video s přidáním objektovým sledovacím subsystémem. Systém byl testován na testovací sadě o 146 obrazech a sekvenční video o trvání 98 minut zahrnujících 128 dopravních značek.

Tabulka 4: Výsledek experimentu aplikace pro statický obraz [15]

	Počet	Procenta
Celkový počet značek	146	100%
Správně rozpoznáno	106	72.6%
Detekční chyba	22	15.1%
Klasifikační chyba	10	13%

Tabulka 5: Výsledek experimentu aplikace pro video [15]

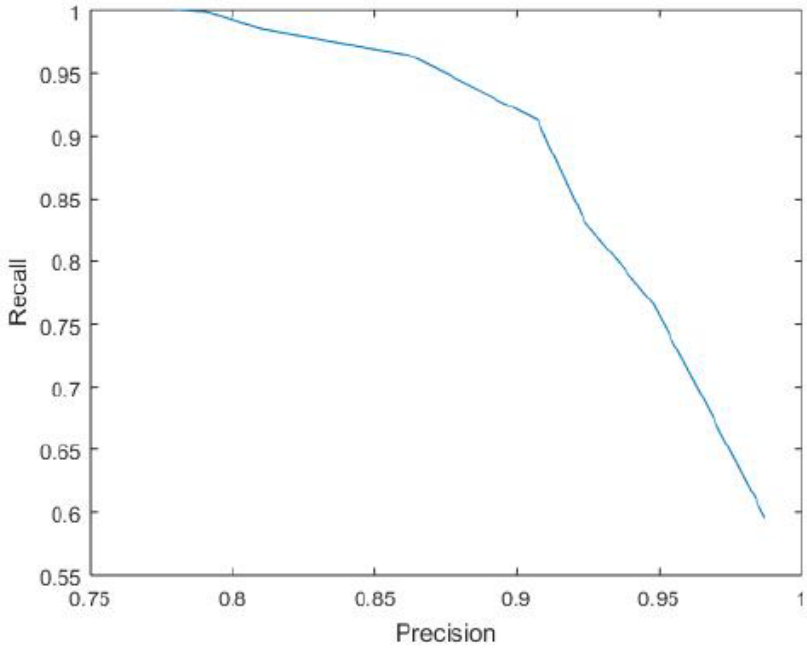
	Počet	Procenta
Celkový počet značek	128	100%
Správně rozpoznáno	106	82.8%
Detekční chyba	13	10.16%
Klasifikační chyba značek	7	4.6%
Klasifikační chyba rychlostních limitů	2	1.56%
Falešně pozitivní	20	

4.2 Traffic Sign Detection and Recognition for Driving Assistance System

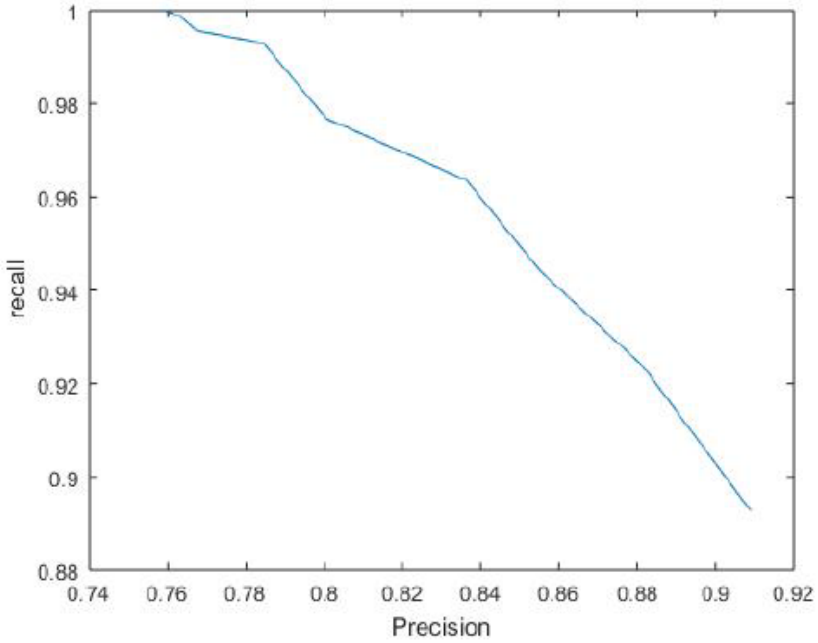
Další řešení přináší článek publikovaný v červnu 2018, na kterém se podílel Huei-Yung Lin, Chin-Chen Chang a Shu-Chun Huang z National Chung Cheng University v Taiwanu. [16] Tato práce se zaměřuje na detekci a klasifikaci dopravního značení pro asistenční systémy při řízení. Systém pro detekci využívá barevné filtry a selektivní vyhledávání. Je detekován velký počet navržených oblastí, které jsou zasílány do konvoluční neuronové sítě pro závěrečnou klasifikaci.

V systému byl zvolen HSI pro posuzování barev. Důvodem bylo, že používá pouze jeden kanál, který reprezentuje interval barev mezi hodnotami 0 až 360 stupňů. Barevný model HSI filtruje většinu regionů, proto muže být omezeno množství navržených oblastí selektivním vyhledáváním. Využily se dva druhy struktur hlubokého učení pro trénování systému. Jednou ze struktur je Alexnet. Tato struktura byla použita z důvodu, že nejen zlepšuje rychlost a přesnost, ale také snižuje problem over-fittingu. Druhou strukturou je GoogLeNet, která v případě velkého množství dat zlepšuje výkon sítě a zvyšuje její šířku a hloubku.

Byly použity dvě rozdílné datové sady jedna pro detekci a druhá pro klasifikaci. Kde pro detekci byla použita datová sada German Traffic Sign Detection Benchmark (GTSDB) a pro klasifikaci German Traffic Sign Recognition Benchmark (GTSRB). Pro klasifikační část byl použit framework Caffe. S použitím 39209 trénovacích obrázku z GTSRB jako trénovací data, která byla rozdělena do 43 typů značek. Testovací datová sada obsahovala 1000 náhodně vybraných obrázků z GTSRB. Pro GoogLeNet byly použity parametry: základní rychlost učení nastavenou na 0,01, počet iterací na 100 000 iterací a útlum váhy na 0,0002. Pro AlexNet byly použity parametry: základní rychlost učení nastavenou na 0,01, počet iterací na 600 iterací a útlum váhy na 0,0005. V článku je také uvedeno, že při více než 600 iteracích, vzniká problém over-fittingu.



Obrázek 12: Výsledky testování GoogLeNet. [16]



Obrázek 13: Výsledky testování AlexNet. [16]

5 Využité technologie a nástroje

Při volbě frameworku pro hluboké učení, jsem zvolil Tensorflow a to z důvodu, že se aktuálně jedná o nejpoužívanější framework. Tím pádem, poskytuje dobrou podporu při problémech. Také má poměrně velkou komunitu, která poskytuje řadu návodů, hotových řešení a rad pro práci s tímto frameworkem.

5.1 Tensorflow

TensorFlow je open source softwarová knihovna pro vysoce výkonné numerické výpočty. Flexibilní architektura umožňuje snadné nasazení výpočtů na různých platformách (CPU,GPU,TPU) a s podporou více CPU a GPU. Byl původně vyvinut výzkumníky a inženýry z Google Brain týmu v rámci organizace Google AI, přichází se silnou podporou pro strojové učení a hlubokého učení s flexibilním numerickým výpočetním jádrem se používá dnes v mnoha dalších vědeckých oblastech. Podporuje vývoj v C++ a Pythonu [8].

5.2 Rozhraní pro detekci objektu (Object detection API)

Hlavním cílem v počítačovém vidění je vytváření přesných modelů strojového učení, které dokážou identifikovat, ale i lokalizovat více objektů v jednom obraze. Tensorflow Objection Detection API je open source framework postavený na Tensorflow, který usnadňuje konstrukci, trénování a nasazení modelu na detekci objektů [9]. Rozhraní pro detekci objektu, nabízí řadu již předem vyškolených modelů na datových sadách COCO, Kitti, Open Images, AVA v2.1 a iNaturalist Species Detection. Tyto předem vyškolené modely mohou být užitečné pro tvorbu modelu z vlastní datové sady [10].

Tabulka 6: COCO trénované modely, nejsou zde uvedeny všechny, pouze modely, které jsem použil pro experimenty ve své práci [10].

Název modelu	Rychlost (ms)	COCO mAP	Výstup
ssd_mobilenet_v2_coco	31	22	Boxy
ssd_inception_v2_coco	42	24	Boxy
faster_rcnn_inception_v2_coco	58	28	Boxy
faster_rcnn_resnet50_coco	89	30	Boxy
rfcn_resnet101_coco	92	30	Boxy

V tabulce nejsou všechny předem vyškolené modely, ale pouze modely, které jsem použil pro experiment ve své práci.

- Rychlost modelu, je zde uvedená v milisekundách pro obraz o velikosti 600×600 pixelů, ale také musíme vzít v potaz, že čas je ovlivněn typem hardwaru. Uvedený čas byl stanoven pomocí grafické karty Nvidia GeForce GTX TITAN X [10].

- Přesnost detektoru je určena pomocí mAP, která je specifická pro datovou sadu COCO nebo Open Images. Pro mAP platí, že vyšší je lepší [10].
- Výstup, znamená jak bude určitý objekt detekován. Jsou zde dvě varianty, první je boxem, tedy čtyřúhelníkem ve kterém je celý detekovaný objekt. Druhá varianta je maska, ta označuje pixely, které obsahují detekovaný objekt. V mém případě jsem použil všechny výstupy typu box.

5.3 Instalace Tensorflow a Object detection API

V téhle práce, jsem vyzkoušel instalaci Tensorflow (1.7.0 - 1.12.0) s podporou grafické karty pro Windows 10 a Linux Ubuntu 18.04.2, který běží na školním serveru merlin1. Při instalaci na merlin1, jsem se setkal s problémem kompatibility verze CUDA a cuDNN, která byla řešená sestavením Tensorflow se zdroje, kde je možnost úpravy konfigurace pro sestavení. Pro sestavení Tensorflow byl použit bazel, kde proces sestavení trval poměrně dlouho. Poté byl sestavený Tensorflow instalován pomocí pip.

Tabulka 7: Seznam podporovaných verzi Python, Bazel, cuDNN a CUDA pro Tensorflow 1.7 - 1.12

Verze Tensorflow	Python verze	Nástroj pro sestavení	cuDNN	CUDA
tensorflow_gpu-1.12.0	2.7, 3.3-3.6	Bazel 0.15.0	7	9
tensorflow_gpu-1.11.0	2.7, 3.3-3.6	Bazel 0.15.0	7	9
tensorflow_gpu-1.10.0	2.7, 3.3-3.6	Bazel 0.15.0	7	9
tensorflow_gpu-1.9.0	2.7, 3.3-3.6	Bazel 0.11.0	7	9
tensorflow_gpu-1.8.0	2.7, 3.3-3.6	Bazel 0.10.0	7	9
tensorflow_gpu-1.7.0	2.7, 3.3-3.6	Bazel 0.9.0	7	9

Chceme-li zjistit, zda instalace proběhla správně a grafická karta je přiřazena, můžeme spustit následující skript.

```
import tensorflow as tf

a = tf.constant([1.0,2.0,3.0,4.0,5.0,6.0],shape=[2,3],name = 'a')
b = tf.constant([1.0,2.0,3.0,4.0,5.0,6.0],shape=[3,2], name = 'b')
c = tf.matmul(a,b)

sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

print(sess.run(c))
```

Výpis 1: tensorflow_testGPU.py

kde výstup může vypadat následovně:

```
Device mapping:
/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GeForce GTX
    950M, pci bus id: 0000:01:00.0
a: (Const): /job:localhost/replica:0/task:0/device:GPU:0
b: (Const): /job:localhost/replica:0/task:0/device:GPU:0
[[22. 28.]
 [49. 64.]]
```

Výpis 2: Výstup po spuštění: tensorflow_testGPU.py

Postup a seznam potřebných balíčku pro instalace Tensorflow s rozhraním pro detekci objektů je dostupný v repozitáři Tensorflow na GitHubu s cestou `models/research/object_detection/g3doc/installation.md`.

5.4 Datové sady

Před vytvářením detektoru objektů z obrazu, je potřeba získat data, které budou použity pro trénování a testování. Chceme-li trénovat robustní klasifikátor, potřebujeme velké množství dat, které se budou lišit. Měli by obsahovat různá pozadí, náhodné objekty, či rozdílné světelné podmínky. Musíme vzít v úvahu, že kvalita této sady může zásadně ovlivnit kvalitu našeho modelu.

Trénovací datová sada pro trénovaní detektoru značek je složena z 6408 jpg snímků obsahující 8628 značek, které byly shromážděný pomocí fotek z Google Street View, snímků získaných s videa a augmentovaných snímků.

Tabulka 8: Seznam značek s počtem výskytů v trénovací datové sadě

Název značky	Počet	Název značky	Počet
Dej přednost v jízdě	1007	Slepá ulice	266
Zákaz vjezdu všech vozidel	441	Jednosměrný provoz	383
Zákaz vjezdu v obou směrech	214	Přechod pro chodce	532
Stop	289	Kruhový objezd	228
Zákaz odbočení vlevo	177	30	258
Zákaz odbočeni vpravo	188	40	309
Zákaz stání	516	50	182
Zákaz zastavení	334	60	130
Hlavni pozemní komunikace	797	70	181
Konec hlavní pozemní komunikace	273	80	209
Křižovatka s vedlejší pozemní komunikací	269	Přikázaný směr jízdy vlevo	169
Pozor zpomalovací práh	322	Přikázaný směr jízdy vpravo	398
Parkoviště	546		

5.4.1 Augmentace dat

K tomu abychom získali kvalitní model, je potřeba velké množství dat. Augmentace je proces, který nám pomáhá s umělým rozšířením data. Využívá se ve strojovém učení, a především v oboru počítačového vidění.

Augmentace je založená na tom, že skutečný snímek, pomocí různých technik transformujeme a tím se nám vygeneruje N nových nový snímků. Každý snímek zobrazuje stále stejný objekt, nicméně jednotlivé snímky jsou náhodně otáčeny, posouvány, přibližovány, oddalovány, je zde přidán jas nebo kontrast, či jiné transformační techniky.

Vždy musíme zvážit jaké transformační techniky můžeme využít. Pokud budeme trénovat neuronovou síť například pro rozpoznávání aut, ztrácí smysl vertikálně otáčet snímky, jelikož neuronová sít se nikdy s takovými daty nesetká. V mém případě pro neuronovou sít na rozpoznávání dopravních značek, jsem nemohl využít, ani horizontální ani vertikální rotaci. Horizontální rotace by ovlivnila značky určující směry vlevo a vpravo.

Obrázek 14: Augmentace horizontální a vertikální rotace značky.

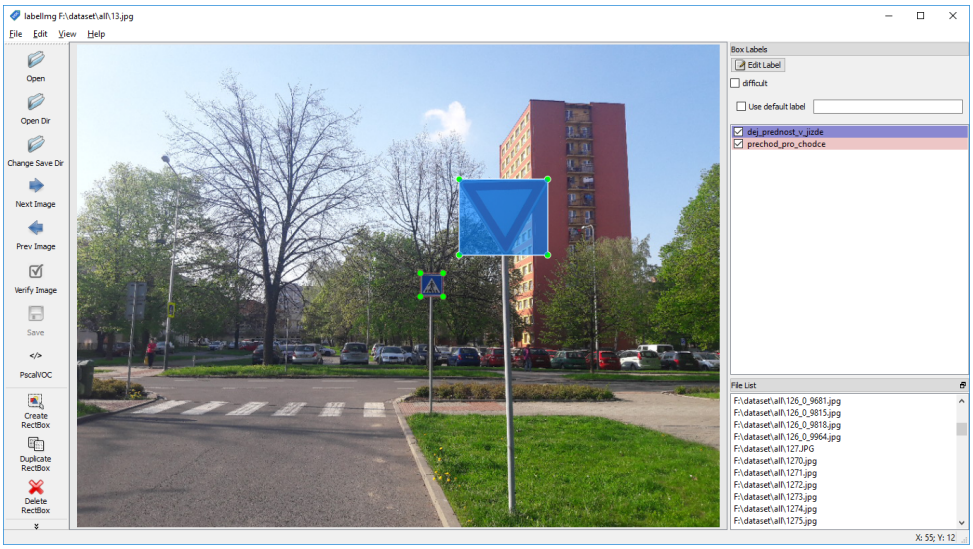
5.5 LabelImg

LabelImg je grafický nástroj, díky kterému lze obrázky velmi snadno anotovat. LabelImg je volně dostupný na `github.com/tzutalin/labelImg`. Je možné si před spuštěním nástroje nastavit seznam tříd, které budeme chtít používat pro anotace, třídy zapíšeme v souboru `/data/predefined_classes.txt`.

Práce s nástrojem LabelImg funguje tak, že se načtou všechny obrázky, které chceme anotovat a následně jim vybereme adresář, do kterého budeme chtít ukládat XML soubor. Tlačítkem "Create RectBox", vybereme ohraničující oblast a přidělíme ji příslušnou třídu. Po uložení je vytvořen příslušný XML soubor ve formátu PACSAL VOC viz. výpis 1. Jedná se o velmi intuitivní nástroj, viz grafické rozhraní na obrázku 7.

```
...
<filename>13.jpg</filename>
...
<object>
      <name>dej_prednost_v_jizde</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
      <bndbox>
         <xmin>571</xmin>
         <ymin>201</ymin>
         <xmax>703</xmax>
         <ymax>314</ymax>
      </bndbox>
</object>
...
```

Výpis 3: Jeden z výstupů programu v PASCAL VOC formátu.



Obrázek 15: Grafický nástroj LabelImg

6 Vytvoření modelu a zhodnocení výsledků

V této kapitole bude stručně vysvětleno, jak připravit data a následně přetrénovat model. V poslední části této kapitoly, budou porovnány výsledné přetrénované modely se zaměřením na přesnost detekce.

6.1 Příprava dat

Pro trénování je potřeba vytvořit vstupní trénovací a testovací soubory dat, které budou převedeny do doporučeného formátu vstupních dat TFrecord. V tomto formátu jsou data uložena binárně a jsou optimalizována několika způsoby pro použití s Tensorflow.

Je potřeba vytvořit značkovací mapu s příponou `.pbtxt`, kde jsou namapovány všechny třídy a jim odpovídající ID.

```
item {
  id: 1
  name: 'dej_prednost_v_jizde'
  display_name: 'Dej prednost v jizde'
}

item {
  id: 2
  name: 'zakaz_vjezdu_vsech_vozidel'
  display_name: 'Zakaz vjezdu vsech vozidel'
}
```

Výpis 4: Ukázka značkovací mapy

6.2 Spuštění trénování

Před spuštěním přetrénování modelu, je potřeba ještě upravit konfigurační soubor, kde je potřeba specifikovat jaký model bude při přetrénování použit, uvést cesty vstupních trénovacích a testovacích dat. Rovněž je také potřeba nastavit cestu pro značkovací mapu, upravit počet tříd pro model, stanovit počet kroků a popřípadě přidat nebo upravit nastavení v `data_augmentation_options`.

V průběh trénování, nás konzole informuje o počtu provedených kroků, jaké v daném kroku byly ztráty a o čas vykonání daného kroku. Doporučený nejnižší limit počtu kroků je 200 tisíc. To však nemusí být vždy dostačující. Ideálním připadám je, pokud se při ztrátách dostaneme trvale pod 0,05.

V průběh trénování jsou tvořeny kontrolní body, neboli bod uložení dosavadních hodnot. Tento průběh trénování je možné sledovat v konzoli nebo pomocí vizualizačního nástroje TensorBoard. TensorBoard je sada webových aplikací pro kontrolu a pochopení běhů a grafů v TensorFlow.

```
INFO:tensorflow:global step 151274: loss = 0.0201 (0.672 sec/step)
INFO:tensorflow:global step 151275: loss = 0.0258 (0.547 sec/step)
INFO:tensorflow:global step 151276: loss = 0.0296 (0.531 sec/step)
INFO:tensorflow:global step 151277: loss = 0.0332 (0.784 sec/step)
INFO:tensorflow:global step 151278: loss = 0.0267 (0.594 sec/step)
INFO:tensorflow:global step 151279: loss = 0.0545 (0.663 sec/step)
INFO:tensorflow:global step 151280: loss = 0.0543 (0.709 sec/step)
INFO:tensorflow:global step 151281: loss = 0.0336 (0.782 sec/step)
INFO:tensorflow:global step 151282: loss = 0.0640 (0.524 sec/step)
INFO:tensorflow:global step 151283: loss = 0.0781 (0.531 sec/step)
INFO:tensorflow:global step 151284: loss = 0.0463 (0.518 sec/step)
INFO:tensorflow:global step 151285: loss = 0.0253 (0.531 sec/step)
INFO:tensorflow:global step 151286: loss = 0.0460 (0.531 sec/step)
```

Výpis 5: Informace o průběhu trénování v konzoli

Po dokončení, je potřeba převod posledního kontrolní bodu a vygenerovat zmrazený inferenční graf (frozen inference graph).

6.3 Výsledky modelů

Pro porovnání přesnosti byly použity metriky Precision, Recall a F1. Pro výpočet těchto metrik je potřeba rozdělit výsledek detekce do 4 stavů [17]:

- Pravdivě pozitivní (TP) = stav, ve kterém model správně předpověděl pozitivní třídu. Pro příklad model vyhodnotí, že konkrétní e-mailová zpráva je spam a e-mailová zpráva je skutečně spam.
- Falešně pozitivní (FP) = stav, ve kterém model nesprávně předpověděl pozitivní třídu. Pro příklad model vyhodnotí, že konkrétní e-mailová zpráva je spam, ale tato e-mailová zpráva ve skutečnosti není spam.
- Pravdivě negativní (TN) = stav, ve kterém model správně předpověděl negativní třídu. Pro příklad model vyhodnotí, že konkrétní e-mailová zpráva není spam a e-mailová zpráva ve skutečnosti není spam.

- Falešně negativní (FN) = stav, ve kterém model nesprávně předpověděl negativní třídu. Pro příklad model vyhodnotí, že konkrétní e-mailová zpráva není spam, ale tato e-mailová zpráva je ve skutečnosti spam.

$$Presicion = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2\frac{Presicion * Recall}{Presicion + Recall} \tag{4}$$

Jak bylo zmíněno bylo přetrénováno pět detekčních sítí Faster R-CNN Inception v2, Faster R-CNN ResNet50, R-FCN ResNet101, SSD Inception v2 a SSD Mobilnet v2. Přetrénování proběhlo za podobných podmínek, což znamená stejné datové sady a stejný počet kroků při trénování, který byl nastaven na 200000.

6.4 Výsledky přetrénovaných modelů pro dopravní značení

Výsledné testování bylo provedeno na nezávislém souboru dat, který byl složen z 154 obrazů o velikosti 1236 × 664 pixelů. Nad těmito obrazy byly testovány jednotlivé detekční sítě, které vyobrazily ohraničující boxy, pokud bylo skóre klasifikace větší než 50%. Všechny modely byly trénovány na školním serveru merlin1, který disponuje dvěma grafickými kartami NVIDIA GeForce RTX 2080.

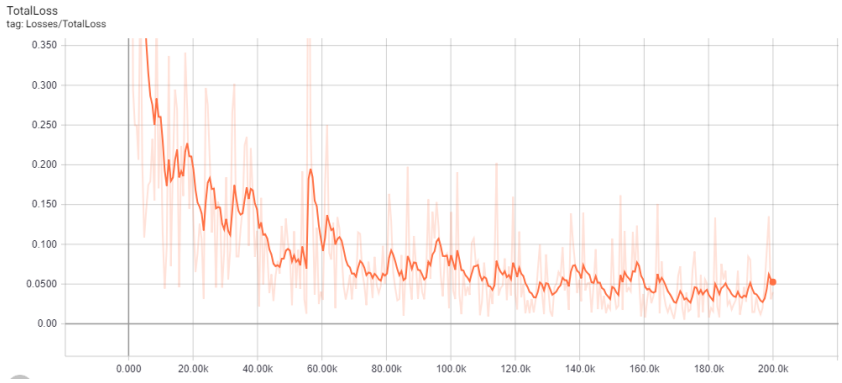
6.4.1 Faster R-CNN Inception v2

Model Faster R-CNN Inception v2 dosahoval poměrně dobrých detekčních i klasifikačních výsledků. Nicméně, jak již bylo zmíněno, tento model není schopen rozpoznat dopravní značení v reálném čase. Doba trénování modelu trvala 10 hodin a 27 minut a celkové ztráty klesly pod 0,05.

Tabulka 9: Výsledky testování Faster R-CNN Inception v2

Název modelu	Faster R-CNN Inception V2
Pravdivě pozitivní	72,1%
Falešně pozitivní	12,4%
Falešně negativní	15,5%
Precision	0,853
Recall	0,823
F1	0,838

Při testování jsem si povšiml, že u dopravního značení dej přednost v jízdě, zákaz stání, zákaz zastavení, hlavní pozemní komunikace, konec hlavní pozemní komunikace, křižovatka s vedlejší



Obrázek 16: Tensorboard TotalLoss s vyhlazením 0,8 pro model Faster R-CNN Inception v2

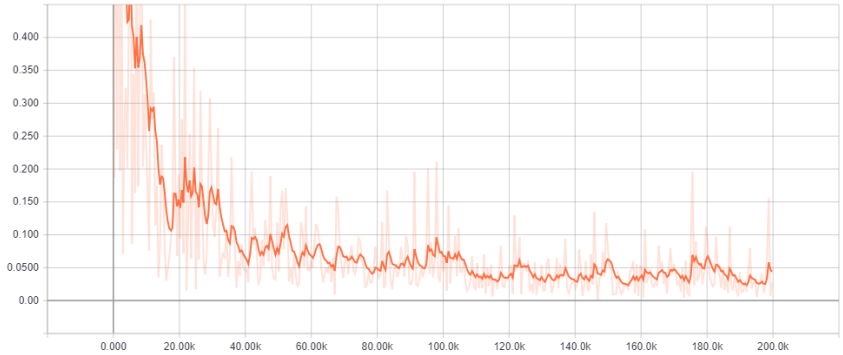
pozemní komunikaci a značky omezující rychlost, měli téměř bezchybnou klasifikaci, zatímco přikázaný směr jízdy vlevo a vpravo a zákaz vjezdu všech vozidel, vykazoval časté klasifikační chyby.

6.4.2 Faster R-CNN ResNet50

Model Faster R-CNN ResNet50 vykazoval o trochu horších klasifikační a detekční výsledky a také je podle zdroje pomalejší než Faster R-CNN Inception v2. Doba trénování modelu trvala 11 hodin a 52 minut a celkové ztráty klesly pod 0,05.

Tabulka 10: Výsledky testování Faster R-CNN ResNet50

Název modelu	Faster R-CNN ResNet50
Pravdivě pozitivní	66,8%
Falešně pozitivní	13,6%
Falešně negativní	19,6%
Precision	0,831
Recall	0,773
F1	0,801



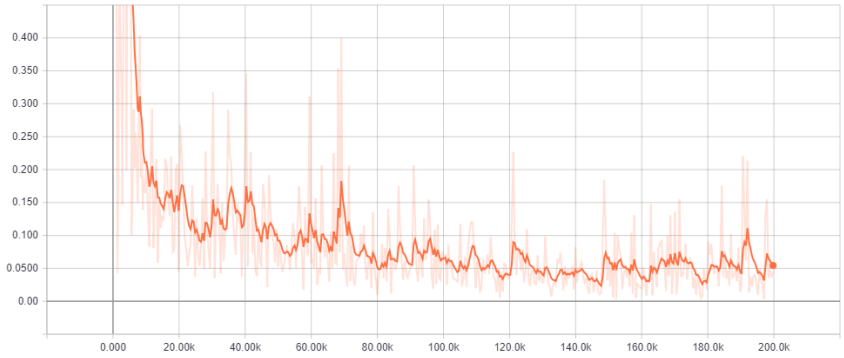
Obrázek 17: Tensorboard TotalLoss s vyhlazením 0,8 pro model Faster R-CNN ResNet50

6.4.3 R-FCN ResNet101

Model R-FCN ResNet101 vykazoval ve výsledku nejlepší detekční a klasifikační výsledky, nicméně jak je podle zdroje uvedeno, jedná se o nejpomalejší model. Doba trénování modelu trvala 14 hodin a 59 minut a celkové ztráty se pohybovaly okolo 0,05.

Tabulka 11: Výsledky testování Faster R-CNN ResNet50

Název modelu	R-FCN ResNet101
Pravdivě pozitivní	77,3%
Falešně pozitivní	12,7%
Falešně negativní	10%
Precision	0,857
Recall	0,883
F1	0,871



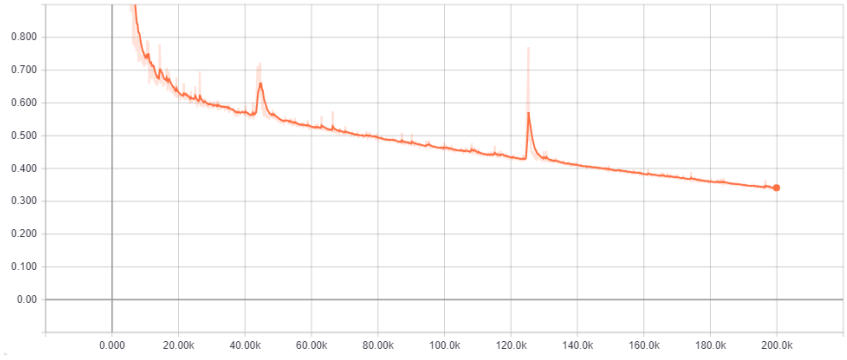
Obrázek 18: Tensorboard TotalLoss s vyhlazením 0,8 pro model R-FCN ResNet101

6.4.4 SSD Inception v2

Model typu SSD Inception v2 měl problém s detekci objektů, avšak velmi dobře klasifikoval dopravní značení. Doba trénování modelu trvala 32 hodin a 36 minut a celkové ztráty se pohybovaly okolo 0,35.

Tabulka 12: Výsledky testování SSD Inception v2

Název modelu	SSD Inception v2
Pravdivě pozitivní	29,6%
Falešně pozitivní	2,4%
Falešně negativní	68%
Precision	0,923
Recall	0,303
F1	0,455



Obrázek 19: Tensorboard TotalLoss s vyhlazením 0,8 pro model SSD Inception v2

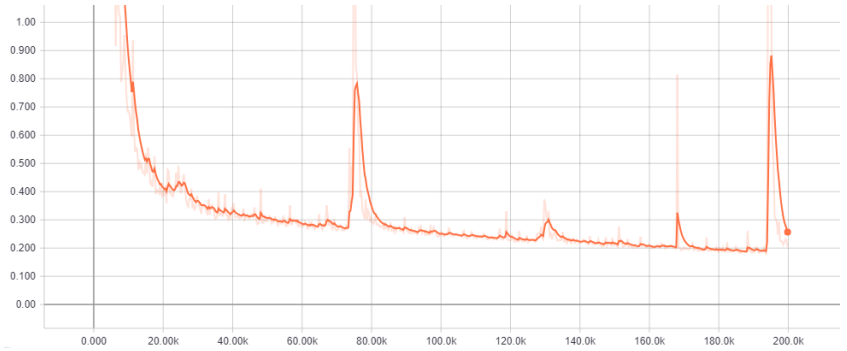
Z grafu je viditelné, že ztráty při trénování neustále klesaly, což by při delším trénování mohlo vést k dosažení lepších výsledků.

6.4.5 SSD Mobilnet v2

Model typu SSD Mobilnet v2 měl stejně jako SSD Inception v2 problém s detekci objektů. Doba trénování modelu trvala 21 hodin a 23 minut a celkové ztráty se pohybovaly okolo 0,35.

Tabulka 13: Výsledky testování SSD Inception v2

Název modelu	SSD Inception v2
Pravdivě pozitivní	30,3%
Falešně pozitivní	6%
Falešně negativní	63,7%
Precision	0,833
Recall	0,322
F1	0,464



Obrázek 20: Tensorboard TotalLoss s vyhlazením 0,8 pro model SSD Mobilnet v2

7 Závěr

Výsledkem této práce bylo navrhnout detektor a klasifikátor dopravního značení v obrazech s využitím frameworku pro strojové učení. Následně model řádně otestovat a zhodnotit výsledky.

Na začátku práce byla vytvořena datová sada pro vytvoření modelu, která se postupem času neustále rozrůstala a výsledně trénovací datová sada obsahovala 6408 snímků s 8628 anotovanými značkami pro 25 typů dopravního značení. Rovněž byla vytvořena nezávislá testovací datová sada pro výsledné testování přesnosti detekčních sítí. Tyto datové sady by bylo vhodné rozšířit dalšími příklady pro některé třídy, nebo popřípadě přidat nové třídy.

Pro vytvoření modelu byl zvolen framework Tensorflow s využitím jeho rozhraním pro detekci objektů, kde práce s tímto rozhraním byla velmi příjemná. Přetrénoval jsem 5 různých druhů detekčních sítí, které byly trénovány na 200 tisíc kroků. Následně byly tyto modely otestovány a porovnány. Z čehož vyplynulo, že nejúspěšnější model je R-FCN ResNet101 s poměrně vysokým skórem F_1 0.87, nicméně tento model je podle zdroje nejpomalejší z vybraných modelů. Pro modely typu SSD, které jsou schopny detekce v reálném čase, by bylo možné zvýšit jejich úspěšnosti, díky delší době trénování.

Chybovost rozpoznávání jednotlivých značek byla odlišná. Jednou z příčin mohl být počet trénovacích vzorků pro jednotlivé třídy. Značka dej přednost v jízdě, měla největší zastoupení v trénovací datové sadě a ve všech modelech dosahovala takřka 100% úspěšnosti. Rovněž značky hlavní pozemní komunikace, zákazu stání, zákazu zastavení, parkoviště a přechodu pro chodce, měly také velmi vysokou úspěšnost, ale zde byla úspěšnost v jednotlivých modelech rozdílná. Všechny tyto značky, až na zákaz zastavení měli v datové sadě více než 500 vzorků. U značek zákaz vjezdu všech vozidel, přikázaný směr jízdy vlevo a přikázaný směr jízdy vpravo byla klasifikace v mnoha případech nepřesná, to mohlo být zapříčiněno podobností těchto značek. Značky kruhový objezd, slepá ulice a stop měly v datové sadě zastoupení méně než 300 vzorků, to se projevilo i na všech modelech, kde tyto značky měly nižší úspěšnost.

Při zpracování této práce jsem nastudoval technologie strojového učení, pro řešení problematiky rozpoznání dopravního značení. Přišlo mi to velmi zajímavé a přínosné, poněvadž mi to dalo možnost seznámení se se strojovým učením, které lze využívat pro mnoho jiných praktických řešení.

Literatura

[1] GERON, Aurelien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-149-1962-299.

[2] Matematická biologie učebnice: Umělá inteligence [online]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence

[3] Středověk umělé inteligence skončil, seznamte se s neuronovými sítěmi, které umí psát básně. Machine Learning Guru [online]. Dostupné z: http://www.mlguru.com/cs/basnik/

[4] A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way [online]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/3bd2b1164a53

[5] Deep Learning for Object Detection: A Comprehensive Review [online]. Dostupné z: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

[6] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [online]. Dostupné z: https://arxiv.org/abs/1506.01497

[7] Understanding Region-based Fully Convolutional Networks (R-FCN) for object detection [online]. Dostupné z: https://medium.com/@jonathan_hui/understanding-region-based-fully-convolutional-networks-r-fcn-for-object_detection-828316f07c99

[8] TensorFlow. [online]. Dostupné z: https://www.tensorflow.org

[9] Tensorflow Object Detection API [online]. Dostupné z: https://github.com/tensorflow/models/blob/master/research/object_detection/README.md

[10] Tensorflow detection model zoo [online]. Dostupné z: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

[11] GIRSHICK, Ross. Fast R-CNN [online]. Dostupné z: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

[12] SSD: LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, Christian SZEGEDY, Scott REED, Cheng-Yang FU a Alexander C. BERG. SSD: Single Shot MultiBox Detector [online]. Dostupné z: https://arxiv.org/pdf/1512.02325.pdf

[13] You Only Look Once: Unified, Real-Time Object Detection [online]. Dostupné z: https://arxiv.org/pdf/1506.02640.pdf

[14] Intersection over Union (IoU) for object detection [online]. Dostupné z: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[15] Real-time detection and recognition of traffic signs [online]. Dostupné z: https://www.researchgate.net/publication/224162933_Real-time_detection_and_recognition_of_traffic_signs

[16] Traffic Sign Detection and Recognition for Driving Assistance System [online]. Dostupné také z: https://www.researchgate.net/publication/327326126

[17] Machine Learning Glossary [online]. Dostupné z: https://developers.google.com/machine-learning/glossary/

Přílohy

- Elektronická verze práce
- Trénovací a testovací datové sady
  - `train_data`
  - `train_labels.csv`
  - `test_data`
  - `label_map.pbtxt`
- Výsledné modely
  - `faster_rcnn_inception_v2/inference_graph`
  - `faster_rcnn_resnet50/inference_graph`
  - `rfcn_resnet101/inference_graph`
  - `ssd_inception_v2/inference_graph`
  - `ssd_mobilenet_v2/inference_graph`
- Utility
  - `calculator_sign.py`
  - `reverse_sign.py`
  - `test_img.py`
  - `test_video.py`
  - `xml_to_csv.py`